2

AD-A228 840

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

3. REPORT TYPE AND DATES COVERED
Annual REport 01 Jun 89- 31 May 90

4. TITLE AND SUBTITLE
Application of Error Detecting/Correcting Codes in Fault Tolerant Logic Design for VLSI

5. FUNDING NUMBERS
F49620-89-C-0069

6. AUTHOR(S)
Professor Lala

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
North Carolina A&T State University
Greensboro NC 27411

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 90 1065

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NE
BLDG 410
BOLLING AFB DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
2305/B1

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

It is now generally accepted that not all faults in VLSI logic can be represented by the stuck-at-0 and stuck-at-1 models used at the gate level. In roder to ensure realistic modeling, faults should be considered at the transistor level, since only at the level the complet circuit structure is known. In other words, test for cirucits should be derived based on possible "shorts" and "opens" at the transistor level. A faculty transistor in a circuit can be modeled as stuck-open or stuck-closed. A stuck-open or stuck-closed stransistor can be modeled by replacing the faulty transistor with an open connection or a direct short respectively between the transistor's source and drain.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASS	UNCLASS	UNCLASS	UL

Annual Report

Project Title :Application of Error Correcting Codes in Fault-tolerant Logic Design for VLSI Circuits

Principal Investigator : Dr. P.K. Lala.

Co-Principal Investigator: Dr. H.L. Martin.

Sponsoring Agency : Air Force Office of Scientific Research

Contractor : North Carolina A&T State University

Contract No. : F49620-89-C-0069

Period of work covered : June 1, 1989 - May 31, 1990

Reporting period: June 1, 1989 - May 31, 1990

In this period, we have completed the following tasks:

i) design r‘ a fault-tolerant universal cell,

ii) development of a methodology for designing self-checking combinational logic circuits.

Fault-tolerant Universal Cell

It is now generally accepted that not all faults in VLSI logic can be represented by the stuck-at-0 and stuck-at-1 models used at the gate level. In order to ensure realistic modeling, faults should be considered at the transistor level, since only at this level the complete circuit structure is known. In other words, tests for circuits should be derived based on possible "shorts" and "opens" at the transistor level.

A faulty transistor in a circuit can be modeled as stuck-open or stuck-closed. A stuck-open or stuck-closed transistor can be modeled by replacing the faulty transistor with an open connection or a direct short respectively between the transistor's source and drain.

The major problem with stuck-open fault is that it forces a combinational circuit to behave as a sequential one. The current strategy for detecting a stuck-open fault is to apply an initializing input pattern, followed by a test pattern that establishes one or more conducting paths from Vdd or ground to the output. However, a two-pattern test can be invalidated by timing skews and charge distribution.

We propose the design of a universal cell constructed from NMOS and PMOS transistors, which is tolerant of stuck-closed or stuck-open fault.The structure of the universal cell is shown in Fig. 1. It is constructed in a way that for each input pattern there are two independent signal paths from the input to the output. The cell can function as one of the conventional two input gate e.g. AND, OR, INVERTER.

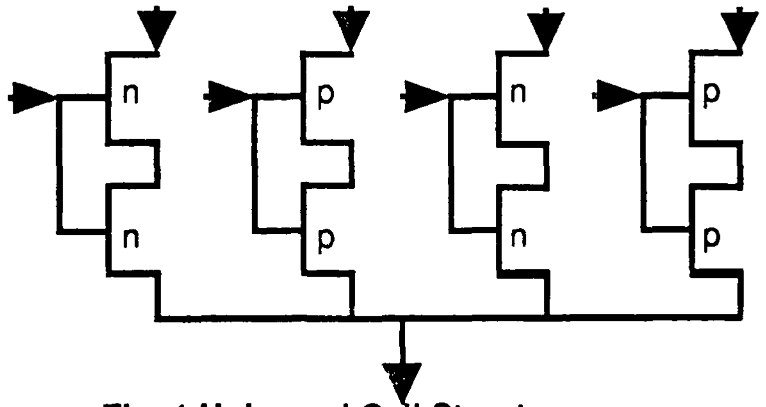

Fig. 1 Universal Cell Structure

The fault-tolerance aspect of the cell are summarized by the following lemmas:

Lemma 1: The universal cell is tolerant of any single stuck-open or stuck-closed fault.

Proof: Every pass transistor in the cell has one transistor in series and another one in parallel. Thus, if a pass transistor is stuck-closed, it will not affect the correct operation of the cell because of the presence of a transistor in series with it. Similarly, if a pass-transistor is stuck-open, its effect will be masked due to the presence of another transistor in parallel with it. Hence any single transistor fault (stuck-open or stuck-closed) will have no effect on the cell. Q.E.D.

Lemma 2: The universal cell is tolerant of multiple stuck-on faults provided there are no more than a single fault in the same path.

Proof: We assumed that each path can only have a single fault; therefore, multiple stuck-on faults are equivalent to single stuck-on faults on different paths. Since all paths are independent, and by Lemma 1 the universal cell is tolerant of single stuck-on faults, the multiple stuck-on faults will have no effect on the correct operation of the cell. Q.E.D.

Lemma 3: The universal cell is tolerant of multiple stuck-open faults provided there are no faults on one of the two signal paths driving the output.

Proof: There are two signal paths to the output for each input combination. If either or both the transistors in one signal path are stuck-open, the path is disconnected from the output. Thus both transistors in the other signal path have to remain fault free in order for the cell to remain operational. Q.E.D.

Logic circuits constructed by using the proposed universal cell will be tolerant of both stuck-closed and stuck-open faults. Currently, as mentioned above, stuck-open faults are detected off-line by applying a two pattern test sequence. In the proposed universal cell, single stuck-open faults will be masked on-line thus eliminating the need for testing such faults. Single stuck-closed in a cell will also have no effect on a circuit during its normal operation.

Self-checking Combinational Circuit Design

Basic Circuitry for Error Detection

The basic block diagram of the self-checking circuits is shown in

Fig 2. It consists of the main combinational circuit, check bit function (additional hardware), check bit generator, and a comparator. The design is based on low-Cost Residue Code (modulo 3). A residue code is called a low-cost residue code if the modulus m can be written as: $m = 2^p - 1$, $p \geq 2$ where p is the number of checkbits.

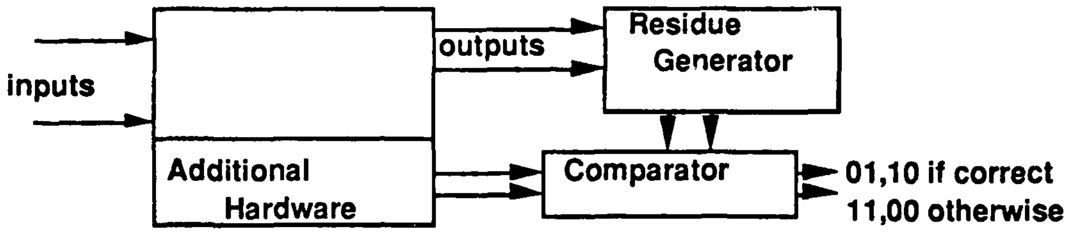

Fig.2
Block diagram of self-checking circuit

The additional blocks of hardware required to implement mod 3 checking consists,

a) A mod 3 residue generator, which produces the appropriate residue for a given data word. It has been shown that a Mod 3 residue generator can be done in three level gates.

b) A two-rail checker which compares the actual and generated residue for a given operation. The design of the two rail checker can be found in [6].

c) Additional hardware part is needed for the calculation of the residue from the inputs directly. This residue does not depend on the hardware in the main block; i.e., any fault in the combinational logic will not affect the residue calculated by using the additional hardware.

Algorithm for Error Detection

Let S be an arbitrary, weighted number system, with X and Y as the inputs. Assume that a single error has occurred in the main combinational circuit operation, and the resultant output is Z'. If Z is the correct output, then by definition, a single error Z' - Z is expressible as:

$$Z' - Z = \pm a_i w_i$$

where a_i and w_i are digits and weights in number system S.

Select a value of m such that $|\pm a_i w_i|_m \neq 0$ for any a_i or w_i. Then,

$$|Z'|_m = |Z \pm a_i w_i|_m$$

If $|\pm a_i w_i|_m \neq 0$ for any a_i and w_i in system S, a single error can be detected by using the following steps :

1- Compute $|Z|_m$ and $|Z'|_m$ from $|X|_m$ and $|Y|_m$.

3- Compare $|Z|_m$ and $|Z'|_m$.

4- If they are not equal, an error has occurred.

Note the restrictive nature of the results. If $|Z|m \neq |Z'|m$, this does not necessarily mean that only a single error has occurred since multiple errors can cause the same effect. Furthermore, if the quantities are equal, this also does not exclude the possibility that an error of multiplicity greater than 1 has occurred.

Rules for Designing the Main Combinational Circuit

A set of rules have been developed to design self-checking circuits based on the principle of the weighted number system. This new set of rules depends on the fan out of every gate; in other words, how many outputs are affected whenever a single fault occurs. It must be noted that this set of rules guarantees on-line detection of single faults in combinational circuits. In order to detect a fault by using low-cost residue code, the outputs, in presence of a fault, have to be changed so that their corresponding residue is different from that of the correct output.

Example 1 :

The fault-free output of a combinational circuit is f3 f2 f1 f0 = 1000.

Since the output is equivalent to decimal 8, its residue is 2 (or 10 in binary). Suppose that a fault affects the output so that

f3 f2 f1 f0 = 1011.

This faulty output has also a residue of 2 which is the same as the residue of the correct output. Hence this fault can not be detected. However, by careful "distribution" of the outputs, this fault can be detected as discussed in the following example.

Example 2:

Consider the same output pattern and the same fault as in example 1. By redistributing the output pattern as

f3 f1 f2 f0 = 1000

the same fault will change the output to

f3 f1 f2 f0 = 1101

The faulty output is equivalent to decimal 13 which has a residue of 1 (or 01 in binary). This residue is different from the original residue, hence the fault is detected.

The above example is straightforward and can be done by intuition. However, this is not the case for more complicated circuits. A set of rules will be presented showing how to distribute the outputs to achieve the desired objective.

Definitions and Basic Concepts

Definition 1: A set of outputs, F, form a cluster if there exists a single fault that can be propagated to all elements of the set at the same time.

Definition 2 : A cluster that has i elements is denoted by Gi.

Definition 3 : Gi is called a subcluster of Gj iff every element in Gi is an element in Gj.

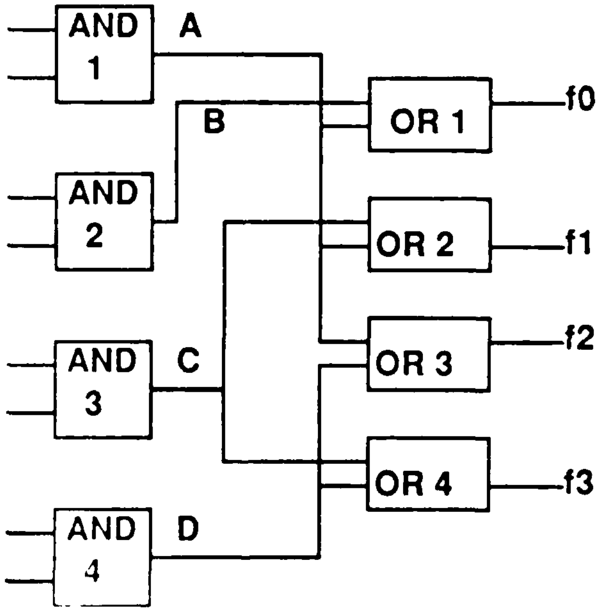

Fig. 3 **Combinational Circuit**

Example 3:

Consider the circuit shown in Fig.3. The outputs f0, f1 and f2 can be in one cluster since a fault at A is propagated to all of them. Similarly, f2 and f3 form a cluster because a fault at D is propagated to both outputs.

Example 4:

In Fig. 3, a fault at D propagates to f2 and f3. Suppose the fault-free output is:

f3 f2 f1 f0 = 0011=3 (equivalent to 0 mod 3.)

If D is stuck-at-1, the output will be

f3 f2 f1 f0 = 1111=15 (equivalent to 0 mod 3.)

The residue of the output does not change because cluster G2=(f2 f3) adds 12, a multiple of three, to the original residue. This is because f2 adds 1 and f3 adds 2 modulo 3. (odd bits contribute 2 and even bits contribute 1 starting from bit # 0.)

<u>Lemma 1</u>

The elements of a cluster Gi have to be distributed in such a way that they will not add up to multiple of 3 in order to get a wrong residue in the presence of a fault.

Proof: Let us assume that a fault create either a single bit error or a uni-directional multi-bit error at the output. Let r be the residue of an output pattern in a fault-free circuit. If a fault affects a certain cluster G, the elements will either change to all 1s or to all 0s, thus adding to or subtracting from the residue a certain number I. The new residue, r', is

$$r' = r \pm \text{residue}(I).$$

If I is a multiple of 3, its residue of I will be zero leaving r unchanged. Thus, the outputs have to be distributed in order to make residue(I) $\neq$ 0, which in turn makes $r \neq r'$. Q.E.D.

<u>Lemma 2</u>:

Any cluster G with a single element i will produce wrong residue if bit i is erroneous at the output.

Proof: If i is erroneous the new reside r' is

$$r' = r \pm \text{residue}(2^i)$$

Depending on the value of i, the residue of 2^i will be either 1 or 2. Consequently, $r' \neq r$. Q.E.D.

We now postulate certain rules based on Lemma 1.

<u>Rule 1</u>

Cluster G2 should have both of its elements located at either odd bit locations or at even bit locations in order to produce wrong residue in the presence of a single fault.

Explanation: Since the fault affects two output bits, the new residue,r', will be the addition (or subtraction) of a number I to the original residue, r. Two odds bits or two even bits will add or subtract to the original residue (2+2) or (1+1) respectively, which are not identical to 0 mod 3. Thus r' will be different from r. On the other hand, if one of the bits is odd and the other is even, then I = (1+2) mod 3 which is equivalent to 0. Hence, r = r' and the fault will not be detected.

Example 5:

In example 4, cluster G2 = (f2 f3) has its elements distributed as one even and one odd. A fault at D does not change the residue. If, however, f2 and f3 are redistributed in the following order f0 f2 f1 f3, the original output will be changed to 1 and in presence of the fault D s-a-1, the residue will be 0.

The following rules can be easily verified.

<u>Rule 2</u>

Cluster G3 should have its elements distributed as two odd and one even or two even and one odd.

Rule 3

Cluster G4 should have its elements distributed as three odd and one even, three even and one odd, four are even or four are odd.

Rule 4

Cluster G5 should have its elements distributed as three odd and two even, three even and two odd, four even and one odd or four odd and one even.

Rule 5

Cluster G6 should have its elements distributed as four even and two odd or four odd and two even.

Rule 6

Cluster G7 should have its elements distributed as four even and three odd or four odd and three even.

Table 3.1 summarizes the distribution of the outputs so that they will produce wrong residue in the presence of a single fault.

Cluster 1	any distribution
Cluster 2	2 even or 2 odd (2e or 2o)
cluster 3	(2e,1o) or (2o,1e)
cluster 4	(3o,1e), (3e,1o), (4o), or (4e)
cluster 5	(3o,2e), (3e,2o), (4e,1o), or (4o,1e)

cluster 6	(4e,2o), (4o,2e)
cluster 7	(4e,3o), (4o,3e)
cluster 8	Distribution is not possible.

Table 3.1 Distribution of outputs in case of fault

Example 6:

In Fig. 3, if A is s-a-1, f1, f2 and f0 will be all logic "1". Consider the following cases:

1. If B = C = D = 1, the fault is masked because f1, f2, and f0 will be 1 in fault free circuit.
2. If B = 0 and D = C = 1, the fault will only propagate to f0; f1 and f2 will remain unchanged. Thus G1 = (f0).
3. If B = C = 0 and D = 1, the fault will propagate to both f1 and f0, leaving f2 unchanged. Thus G2 = (f0 f1)
4. If B = D = C = 0, the fault will propagate to f0, f1 and f2. Thus G3 = (f0 f1 f2).

This example shows that although originally we have G3, one can get Gi's with fewer elements. Therefore, any subcluster Gj can be formed from Gi.

The above statement infers that one does not need to trace the different paths that lead from a fault site to the outputs. On the other hand, depending on the input vector, different subclusters can be formed as illustrated in example 6.

If there is a cluster Gi where $i \geq 3$, rules 1 and rule 2 can not be applied directly. This is because, in such cases if rule 1 is applied, the application of rule 2 will result in contradiction. For example, three different subclusters were formed; i.e., G1 = (f0), G2 = (f0 f1) and G3 = (f0 f1 f2) in Example 6. Two more subclusters (f0 f2), (f1 f2) can be formed when inputs B=D=0, C=1 and C=D=0, B=1 are applied respectively for the same fault. Assume that we have more than four outputs so that we can distribute the three output bits, f0, f1, and f2 to all odd or all even positions after applying rule 1 to the following subclusters (f0 f1), (f0 f2) and (f1 f2).

Since we have G3 = (f1 f2 f0), the above distribution of the outputs contradicts rule 2. Similarly, we will reach contradiction for Gi's with $i > 3$.

<u>Lemma 3</u>

Rule 1 will always be applicable to G2 irrespective of the number of subclusters formed.

Proof: Since G1 is the only subcluster that can be formed from G2, and since rule 1 always guarantees wrong residue in the presence of a fault in the circuit, the application of rule 1 to G2 will always produce wrong residue for a fault that affects two outputs.Q.E.D.

Lemma 4

Any cluster Gi with i $\geq$ 3 can be decomposed into different subclusters G1 and G2.

Proof: Cluster Gi is formed by tracing the paths from the fault site to different outputs. By replicating the faulty gate so that each new gate has paths that lead only to one or two outputs,the resulting clusters are G1 and G2.

Example 7:

In Fig. 3, a fault at the output of AND 1 may affect three outputs f0, f1 and f3; thus G3 = (f0 f1 f2). By duplicating the gate A, G3 can be decomposed into G1 = (f0) and G2 = (f1 f2). As shown in Fig. 4, gate A has been replaced by gates A' and A".

In decomposing Gi (i $\geq$ 3) into subclusters of G2's, there will be less constraint on the implementation of the circuit if groups of G2 are identical.

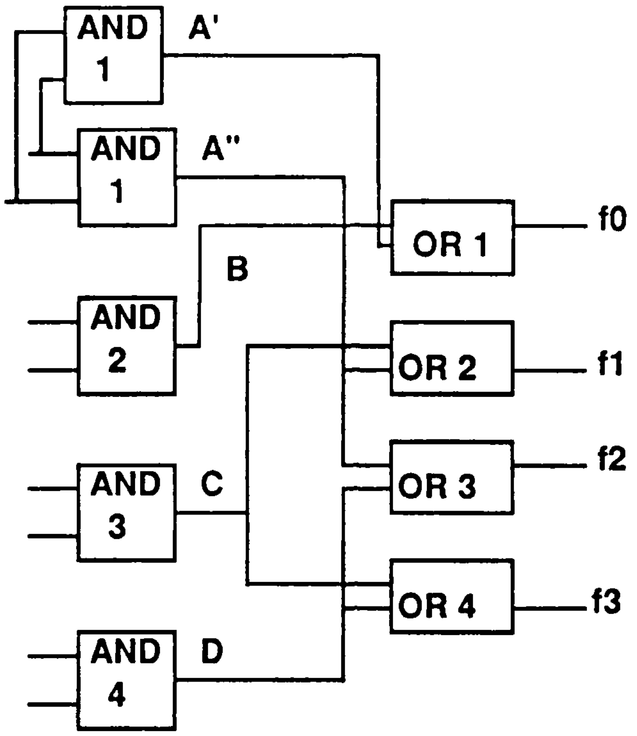

Fig. 4
Modified design of Fig. 3

If it is necessary to have more odd or even bits than that are available, either the number of the outputs should be increased or more gates should be added. The example below shows how to deal with such problems.

Example 8:

Consider a circuit of eight outputs. Let us assume that we get a circuit which has the following clusters, (f0 f1), (f0 f2), (f2 f3), (f2 f4), (f5 f6), (f6 f7)

After applying rule 1, it was found that f0, f1, f2, f3, and f4 have to be even, and f5, f6, and f7 have to be odd. One solution can be as follows:

8	7	6	5	4	3	2	1	0
f4		f3		f2		f1		f0
	0		f7		f6		f5	

In this example, the number of outputs is increased by 1. In nine outputs, there is five even bits and four odd bits. Since only three odd bits are needed, one of the odd bits is set to zero. It should be noted that since the residue of a nine output circuit may be different from that of the original eight output circuit, the additional logic responsible for generating the residue from the inputs have to be designed according to the new distributions of output bits

The other solution is to decompose the cluster (f2 f4) into two subclusters (f4) and (f2). In this case, f0, f1, f2 and f3 have to be even and the other outputs have to be odd.

3.6 Summary of the Rules

A summary for the rules and restrictions needed to design self-checking circuits using low-cost residue code is given below.

1. Avoid using EX-OR or EX-NOR gates in a combinational circuit because they may produce non-unidirectional errors at the output.
2. Make sure that all faults propagate either to one or two outputs only. This can be achieved by duplicating the faulty gate and distributing the fanouts among the duplicated gates. In other words, decompose all clusters into subclusters of G1 and G2.
3. When decomposing different groups into G1 and G2, try to get same G's. This will reduce the size of the circuit.
4. When more even or odd bits are need than what are available, follow the techniques discussed in example 8.